Mais Sobre

Cifras de Bloco

Prof. Dr. Iaçanã Ianiski Weber

Confiabilidade e Segurança de Software

98G08-4

Agradecimentos especiais ao Prof. Avelino Zorzo e aos Autores Christof Paar e Jan Pelzl pelo material.

Índice

Cifras de Bloco

- Uma cifra de bloco é muito mais do que apenas um algoritmo de criptografia, ela pode ser usada para:
  - construir diferentes tipos de esquemas de criptografia baseados em blocos
  - implementar uma cifra de fluxo
  - construir funções hash
  - criar códigos de autenticação de mensagens
  - construir protocolos de estabelecimento de chave
  - criar um gerador de números pseudoaleatórios
  - ...
- A segurança das cifras de bloco também pode ser aumentada por meio de:
  - *key whitening*
  - múltipla encriptação

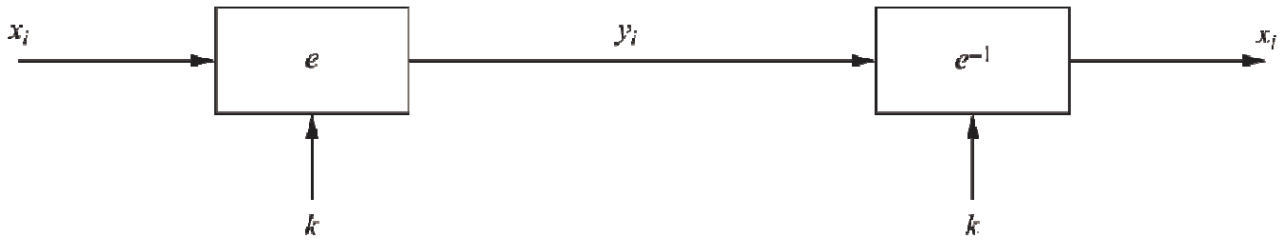
Índice

Criptografia com Cifras de Bloco

- Existem diversas maneiras de criptografar textos longos, por exemplo, um e-mail ou um arquivo de computador, usando uma cifra de bloco (*"modes of operation"*):
  - Modo Electronic Code Book (ECB)
  - Modo Cipher Block Chaining (CBC)
  - Modo Output Feedback (OFB)
  - Modo Cipher Feedback (CFB)
  - Modo Counter (CTR)
  - Modo Galois Counter (GCM)
- Todos os 6 modos têm um objetivo:
  - Além da confidencialidade, eles fornecem autenticidade e integridade:
    - A mensagem realmente veio do remetente original? (autenticidade)
    - O texto cifrado foi alterado durante a transmissão? (integridade)

Modo Electronic Code Book (ECB)

- $e_k(x_i)$ denota a encriptação de um bloco de texto claro x_i de b bits com a chave k
- $e_k^{-1}(y_i)$ denota a decriptação de um bloco de texto cifrado y_i de b bits com a chave k
- Mensagens que excedem b bits são particionadas em blocos de b bits
- **Cada bloco é encriptado separadamente**



Encriptação: $y_i = e_k(x_i), \quad i \geq 1$
Decriptação: $x_i = e_k^{-1}(y_i) = e_k^{-1}(e_k(x_i)), \quad i \geq 1$

ECB: vantagens/desvantagens

- **Vantagens**
  - não é necessária sincronização de blocos entre o remetente e o receptor
  - erros de bits causados por canais ruidosos afetam apenas o bloco correspondente, mas não os blocos seguintes
  - a operação da cifra de bloco pode ser paralelizada
    - vantagem para implementações de alta velocidade
- **Desvantagens**
  - ECB encripta de forma altamente determinística
    - textos claros idênticos resultam em textos cifrados idênticos
    - um atacante pode reconhecer se a mesma mensagem foi enviada duas vezes
  - blocos de texto claro são encriptados independentemente dos blocos anteriores
    - um atacante pode reordenar blocos cifrados, resultando em um texto claro válido

Ataque de Substituição no ECB

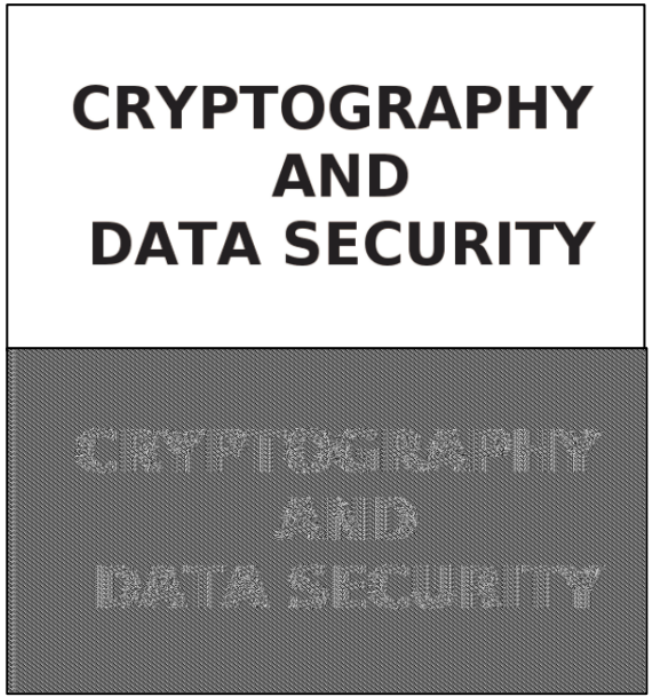
- Uma vez que um mapeamento de um bloco de texto claro x_i para um bloco de texto cifrado y_i é conhecido, uma sequência de blocos cifrados pode ser facilmente manipulada.
- Suponha uma *transferência bancária eletrônica*:

Bloco #	1	2	3	4	5
	Banco A remetente	Conta # remetente	Banco B destinatário	Conta # destinatário	Valor $

- a chave de encriptação entre os dois bancos não muda com muita frequência
- O atacante envia transferências de $1,00 da sua conta no banco A para sua conta no banco B repetidamente
  - Como o conteúdo da transação (remetente, destinatário, valores) não muda entre as transferências, alguns blocos cifrados são idênticos.
  - O atacante identifica quais blocos cifrados correspondem ao campo "conta de destino"(Bloco 4) porque ele sempre envia para a mesma conta.
  - Ele armazena os blocos cifrados 1, 3 e 4, que permanecem os mesmos em todas as transferências.
- Ele intercepta ou modifica transferências de terceiros:
  - Outra pessoa está enviando dinheiro do Banco A para uma conta legítima no Banco B.
  - A transação deles gera os blocos cifrados originais: Bloco 1º, Bloco 2º, Bloco 3º, Bloco 4º, Bloco 5º.
  - O atacante substitui o Bloco 4º (conta de destino legítima) pelo Bloco 4 que ele armazenou antes:
  - Ele cria um novo conjunto: Bloco 1º, Bloco 2º, Bloco 3º, Bloco 4 (dele), Bloco 5º.

Exemplo de criptografia de *bitmaps* no modo ECB

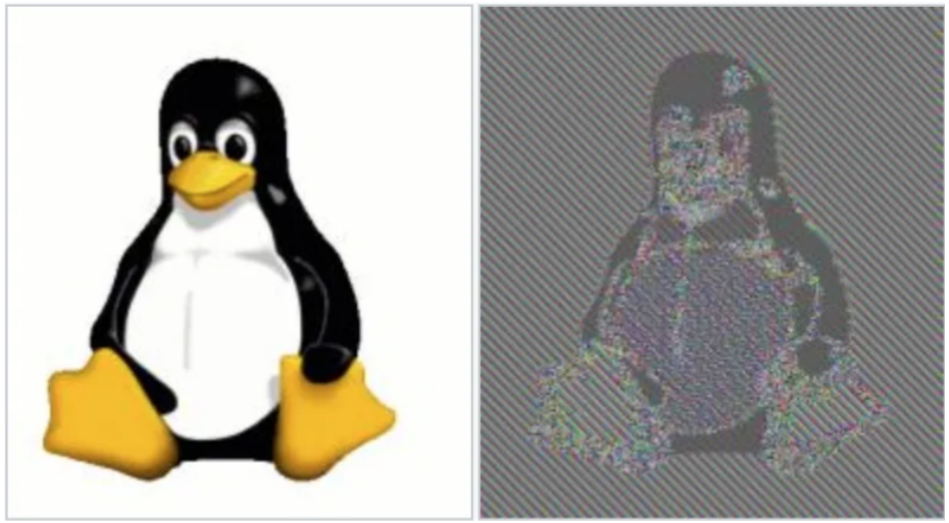
- Textos claros idênticos são mapeados para textos cifrados idênticos



- As propriedades estatísticas do texto claro são preservadas no texto cifrado

Exemplo de criptografia de *bitmaps* no modo ECB

- Textos claros idênticos são mapeados para textos cifrados idênticos



- As propriedades estatísticas do texto claro são preservadas no texto cifrado

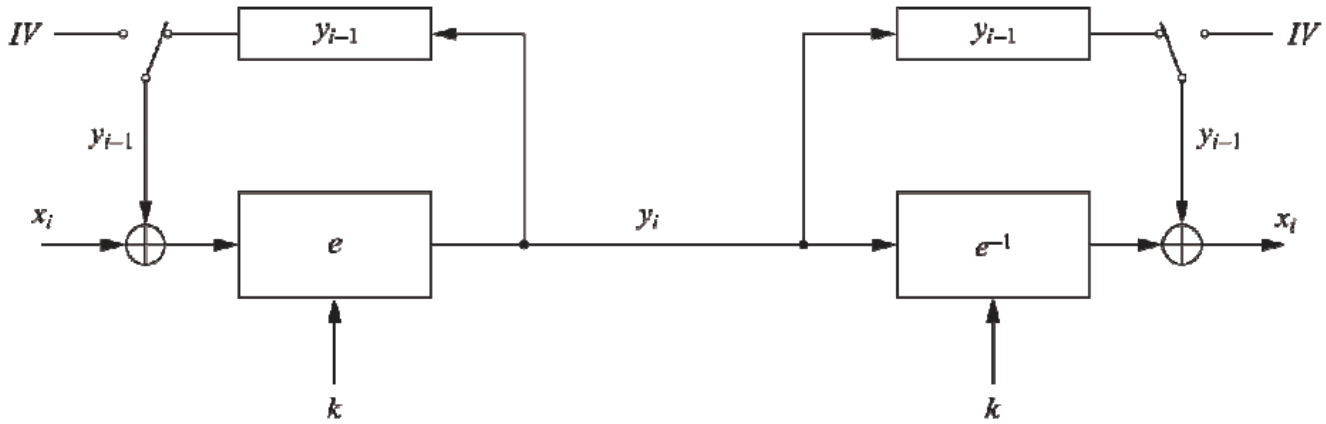
Modo Cipher Block Chaining (CBC)

- Há duas ideias principais por trás do modo CBC:
  - A encriptação de todos os blocos está "encadeada"
    - o texto cifrado y_i depende não apenas do bloco x_i, mas também de todos os blocos de texto claro anteriores
  - A encriptação é randomizada usando um vetor de inicialização (IV)

Encriptação (primeiro bloco): $y_1 = e_k(x_1 \oplus IV)$
Encriptação (blocos gerais): $y_i = e_k(x_i \oplus y_{i-1}), \quad i \geq 2$
Decriptação (primeiro bloco): $x_1 = e_k^{-1}(y_1) \oplus IV$
Decriptação (blocos gerais): $x_i = e_k^{-1}(y_i) \oplus y_{i-1}, \quad i \geq 2$

Modo Cipher Block Chaining (CBC)

- Para o primeiro bloco de texto claro x_1, não existe texto cifrado anterior
  - um IV é adicionado ao primeiro texto claro para tornar cada encriptação CBC não determinística
  - o primeiro texto cifrado y_1 depende do texto claro x_1 e do IV
- O segundo texto cifrado y_2 depende do IV, x_1 e x_2
- O terceiro texto cifrado y_3 depende do IV, x_1, x_2 e x_3, e assim por diante

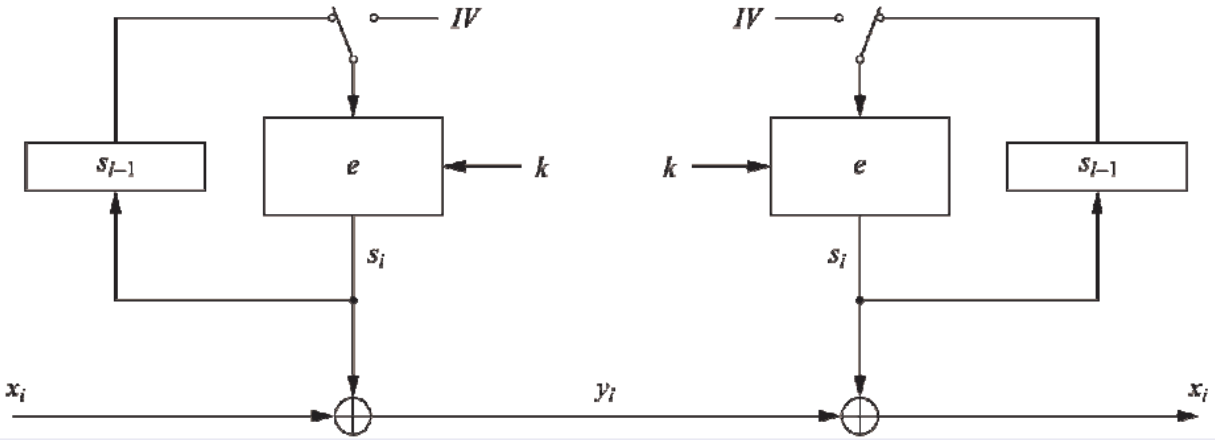


Ataque de Substituição no CBC

- Suponha o exemplo anterior (*transferência bancária eletrônica*)
- Se o IV for escolhido corretamente para cada transferência, o ataque não funcionará
- Se o IV for mantido o mesmo em várias transferências, o atacante poderá reconhecer as transferências da sua conta no banco A para o banco B
- Se escolhermos um novo IV a cada vez que encriptamos, o modo CBC se torna um esquema de encriptação probabilístico, ou seja, duas encriptações do mesmo texto claro parecerão completamente diferentes
- Não é necessário manter o IV *secreto*!
- Normalmente, o IV deve ser um *nonce* público (valor usado apenas uma vez)

Modo Output Feedback (OFB)

- É usado para construir uma *cifra de fluxo síncrona* a partir de uma cifra de bloco
- O *key stream* não é gerado bit a bit, mas sim em blocos
- A saída da cifra nos fornece os bits do *key stream* S_i, com os quais podemos encriptar os bits do texto claro usando a operação XOR



Encriptação (primeiro bloco): $s_1 = e_k(IV)$ e $y_1 = s_1 \oplus x_1$

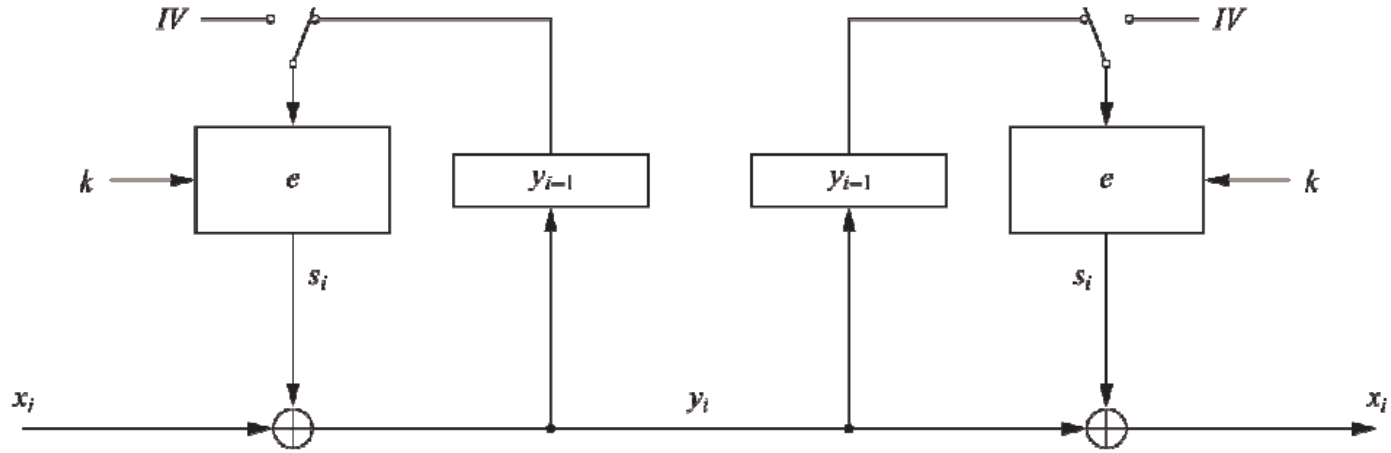
Encriptação (blocos gerais): $s_i = e_k(s_{i-1})$ e $y_i = s_i \oplus x_i, \quad i \geq 2$

Decriptação (primeiro bloco): $s_1 = e_k(IV)$ e $x_1 = s_1 \oplus y_1$

Decriptação (blocos gerais): $s_i = e_k(s_{i-1})$ e $x_i = s_i \oplus y_i, \quad i \geq 2$

Modo Cipher Feedback (CFB)

- Utiliza uma cifra de bloco como base para construir uma *cifra de fluxo assíncrona* (semelhante ao modo OFB)
- O *key stream* S_i é gerado em blocos e também depende do texto cifrado
- Uso de IV faz a encriptação no modo CFB não determinística

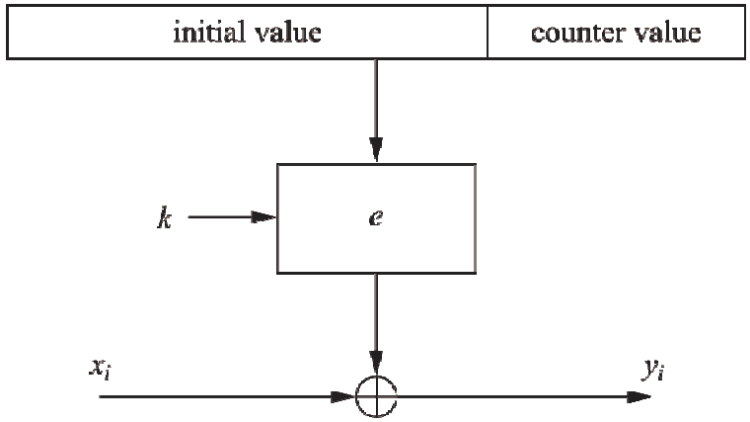


Encriptação (primeiro bloco): $y_1 = e_k(IV) \oplus x_1$
Encriptação (blocos gerais): $y_i = e_k(y_{i-1}) \oplus x_i, \quad i \geq 2$
Decriptação (primeiro bloco): $x_1 = e_k(IV) \oplus y_1$
Decriptação (blocos gerais): $x_i = e_k(y_{i-1}) \oplus y_i, \quad i \geq 2$

Pode ser usado em situações onde blocos curtos de texto claro precisam ser encriptados

Modo Counter (CTR)

- Utiliza uma cifra de bloco como uma *cifra de fluxo* (como os modos OFB e CFB)
- O *key stream* é calculado em blocos
- A entrada da cifra de bloco é um contador, que assume um valor diferente a cada vez que a cifra calcula um novo bloco do *key stream*



- Diferente dos modos CFB e OFB, o modo CTR pode ser paralelizado, pois a segunda encriptação pode começar antes da primeira terminar
  - Desejável para implementações de alta velocidade, por exemplo, em roteadores de rede

Encriptação: $y_i = e_k(IV \,\|\, CTR_i) \oplus x_i, \quad i \geq 1$
Decriptação: $x_i = e_k(IV \,\|\, CTR_i) \oplus y_i, \quad i \geq 1$

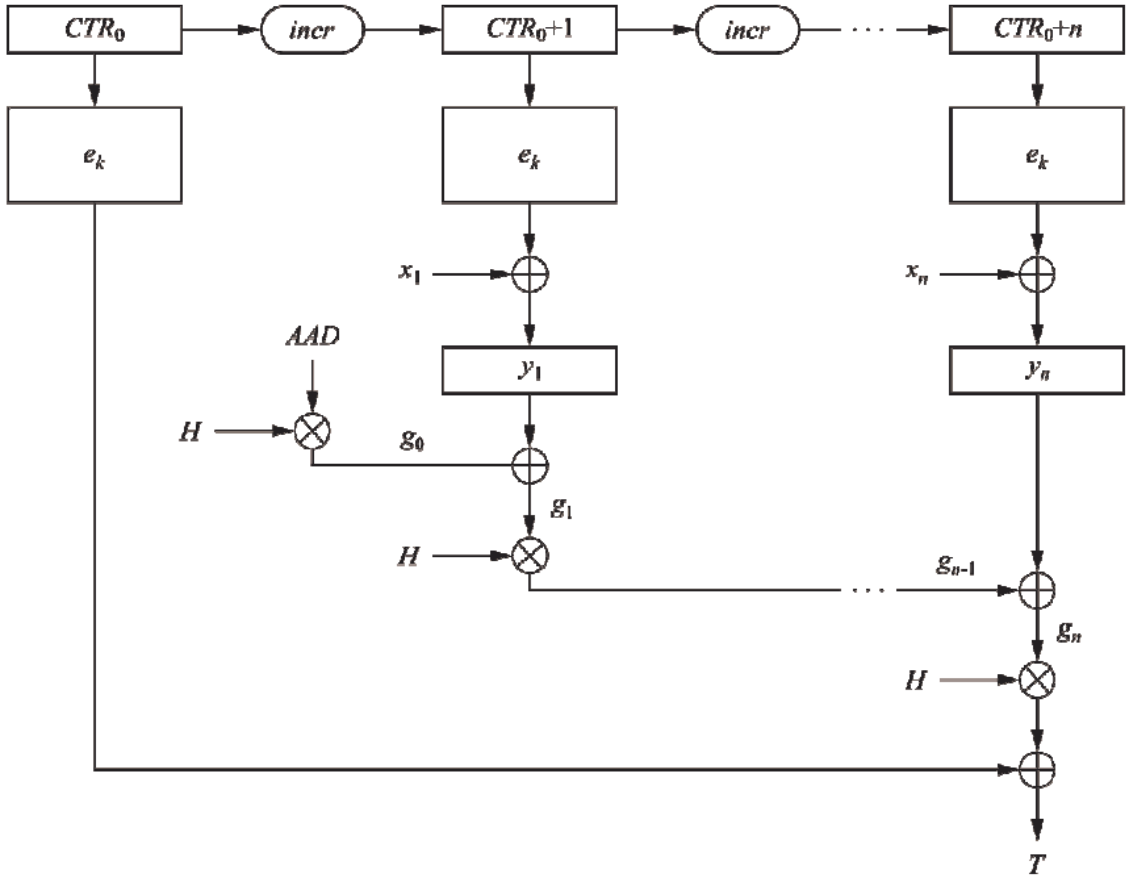
Modo Galois Counter (GCM)

- Também calcula um *código de autenticação de mensagem* (MAC), ou seja, um "*checksum* criptográfico" é calculado para uma mensagem
- Ao utilizar o GCM, dois serviços adicionais são fornecidos:
  - **Autenticação de Mensagem**
    - o receptor pode garantir que a mensagem foi realmente criada pelo remetente original
  - **Integridade da Mensagem**
    - o receptor pode garantir que ninguém alterou o texto cifrado durante a transmissão

Modo Galois Counter (GCM)

- **Para encriptação**
  - Um contador inicial é derivado de um IV e de um número de série
  - O valor inicial do contador é incrementado, então encriptado e aplicado um XOR com o primeiro bloco de texto claro
  - Para os textos claros subsequentes, o contador é incrementado e então encriptado
- **Para autenticação**
  - É realizada uma multiplicação encadeada no campo de Galois
  - Para cada texto claro, é derivado um parâmetro intermediário de autenticação g_i
    - g_i é calculado como o XOR do texto cifrado atual com o último g_{i-1} e multiplicado pela constante H
    - H é gerado pela encriptação da entrada zero com a cifra de bloco
  - Todas as multiplicações são feitas no campo de Galois de 128 bits $GF(2^{128})$

Modo Galois Counter (GCM)



Encriptação:

- Derivar um valor de contador CTR_0 a partir do IV e calcular $CTR_1 = CTR_0 + 1$
- Calcular o texto cifrado: $y_i = e_k(CTR_i) \oplus x_i, \quad i \geq 1$

Autenticação:

- Gerar a subchave de autenticação $H = e_k(0)$
- Calcular $g_0 = AAD \times H$ (multiplicação no campo de Galois)

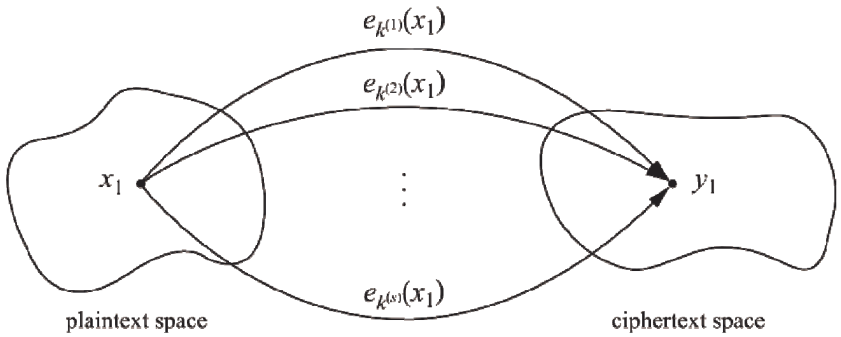
Índice

Busca Exaustiva por Chave Revisitada

- Uma busca exaustiva simples por uma chave DES conhecendo um par (x_1, y_1):

$$DES_{k_i}(x_1) \stackrel{?}{=} y_1, \quad i = 0, 1, \ldots, 2^{56} - 1$$

- No entanto, para a maioria das outras cifras de bloco, a busca por chave é mais complicada
- Um ataque de força bruta pode produzir *falsos positivos*
  - chaves k_i encontradas podem não ser a chave real usada na encriptação



- A probabilidade disso ocorrer está relacionada ao tamanho relativo entre o espaço de chaves e o espaço de textos claros
- Um ataque de força bruta ainda é **possível**, mas requer diversos pares de texto claro–texto cifrado

Um Exemplo de Busca Exaustiva por Chave

- Suponha uma cifra com largura de bloco de 64 bits e tamanho de chave de 80 bits
- Se encriptarmos x_1 com todas as 2^{80} chaves possíveis, obtemos 2^{80} textos cifrados
  - No entanto, existem apenas 2^{64} possíveis diferentes
- Se testarmos todas as chaves para um par texto claro–texto cifrado, encontramos em média:

$$\frac{2^{80}}{2^{64}} = 2^{16} \text{ chaves que realizam o mapeamento } e_k(x_1) = y_1$$

- **Fórmula geral:**

Dada uma cifra de bloco com chave de k bits e bloco de n bits, e t pares texto claro–texto cifrado $(x_1, y_1), \ldots, (x_t, y_t)$, o número esperado de *falsas chaves* que cifram todos os textos claros corretamente é: 2^{k-tn}

- Neste exemplo, assumindo dois pares texto claro–texto cifrado, a probabilidade de falso positivo é:

$$2^{80-2\cdot 64} = 2^{-48}$$

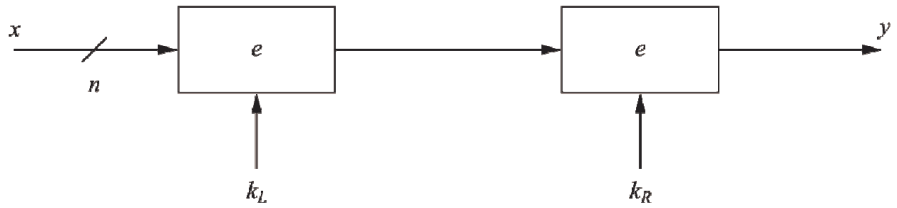
Índice

Aumentando a Segurança de Cifras de Bloco

- Em algumas situações, desejamos aumentar a segurança de cifras de bloco, por exemplo, se uma cifra como o DES estiver disponível em hardware ou software por motivos de legado em uma aplicação específica
- Duas abordagens são possíveis:
  - **Múltiplas encriptações**
    - teoricamente muito mais seguras, mas **às vezes** na prática aumentam muito pouco a segurança
  - **Key whitening**

Dupla Encriptação

- Um plaintext x é primeiro encriptado com uma chave k_L, e o ciphertext resultante é encriptado novamente usando uma segunda chave k_R.



- Assumindo um comprimento de chave de k bits, uma busca exaustiva por chaves exigiria

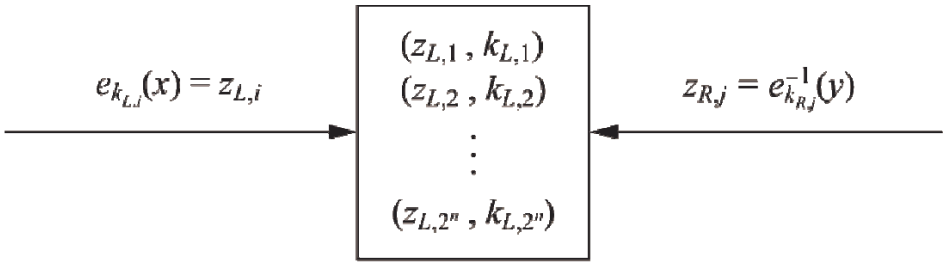
$$2^k \cdot 2^k = 2^{2k}$$

encriptações ou decriptações.

Ataque Meet-in-the-Middle (1)

- Um ataque **Meet-in-the-Middle** requer:

$$2^k + 2^k = 2^{k+1} \text{ operações!}$$



Fase I — Exploração da primeira camada de cifra (k_L):

- Para um par conhecido (x_1, y_1), o atacante testa todas as chaves $k_{L,i}$, com $i = 1, 2, \ldots, 2^k$.
- Para cada chave, calcula:

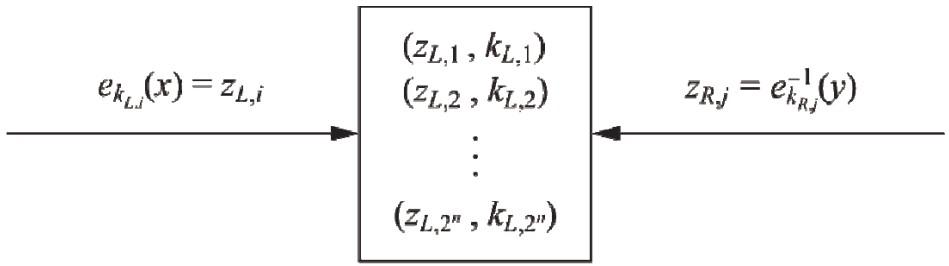
$$z_{L,i} = E_{k_{L,i}}(x_1)$$

- Os valores $z_{L,i}$ e suas chaves associadas são armazenados em uma tabela.
- A tabela é ordenada com base nos valores $z_{L,i}$, facilitando buscas rápidas.

Ataque Meet-in-the-Middle (2)

- Um ataque **Meet-in-the-Middle** requer:

$$2^k + 2^k = 2^{k+1} \text{ operações!}$$



Fase II — Verificação da segunda camada de cifra (k_R):

- O atacante testa todas as chaves $k_{R,j}$ possíveis e calcula:

$$z_{R,j} = D_{k_{R,j}}(y_1)$$

- Verifica se $z_{R,j}$ existe na tabela da Fase I.
- Se houver correspondência, o par de chaves $(k_{L,i}, k_{R,j})$ é um candidato à chave correta.

Ataque Meet-in-the-Middle (3)

Complexidade Computacional:

número de encriptações e decriptações: $2^k + 2^k = 2^{k+1}$
número de posições de armazenamento: 2^k

- Dupla encriptação não é muito mais segura que encriptação simples!

Triple Encryption

- A **encriptação de um bloco três vezes**:

$$y = e_{k_3}\left(e_{k_2}\left(e_{k_1}(x)\right)\right)$$

- Na prática, uma variante chamada **EDE** (enc-dec-enc) é frequentemente usada:

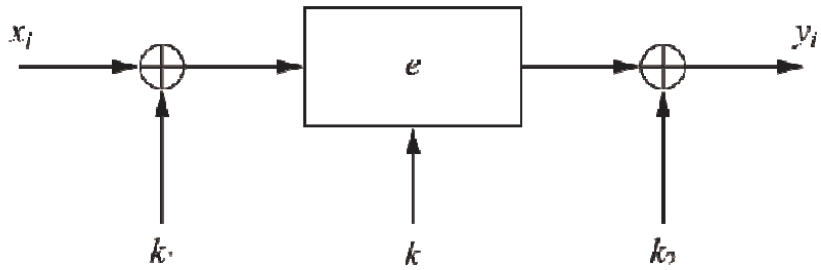
$$y = e_{k_3}\left(e_{k_2}^{-1}\left(e_{k_1}(x)\right)\right)$$

- **Vantagem**: escolhendo $k_1 = k_2 = k_3$ realiza apenas uma encriptação simples.
- Ainda assim, é possível realizar um *meet-in-the-middle attack*, que reduz o tamanho efetivo da chave da *triple encryption* de $3K$ para $2K$

Triple encryption efetivamente dobra o tamanho da key

Key Whitening

- Torna *block ciphers* mais resistentes a ataques de força bruta
- Além da *cipher key* regular k, duas *whitening keys* k_1 e k_2 são usadas para aplicar XOR no *plaintext* e no *ciphertext*



- Não fortalece *block ciphers* contra a maioria dos ataques analíticos, como criptoanálise linear e diferencial
- Não é uma "cura" para cifradores fracos por natureza
- A carga computacional adicional é desprezível
- Sua principal aplicação é em cifradores que são relativamente fortes contra ataques analíticos, mas que possuem um espaço de *key* muito pequeno — especialmente o DES
  - Uma variante do DES que utiliza *key whitening* é chamada **DESX**

Lições Aprendidas

- Existem várias formas de encriptar com um *block cipher*. Cada modo de operação possui vantagens e desvantagens
- Diversos modos transformam um *block cipher* em um *stream cipher*
- Há modos que realizam encriptação juntamente com autenticação, isto é, um *checksum* criptográfico protege contra manipulação da mensagem
- O modo ECB simples possui vulnerabilidades de segurança, independentemente do qual *block cipher* for utilizada
- O modo *counter* permite paralelização da encriptação e, portanto, é adequado para implementações de alta velocidade
- Dupla encriptação com um mesmo *block cipher* apenas melhora marginalmente a resistência contra ataques de força bruta
- Tripla encriptação com um mesmo *block cipher* aproximadamente **dobra** o comprimento da *key*
- O *key whitening* aumenta o comprimento da *key* do DES sem sobrecarga computacional significativa

Índice

Exercício: Modos de Operação com um Cifrador Simples

Considere um *toy block cipher* $e()$ para encriptação de blocos de 5 bits. A função de encriptação é uma permutação de bits, que depende da *key*. Assuma que, para uma dada *key*, a encriptação (permutação) é a seguinte:

$$e(b_1\,b_2\,b_3\,b_4\,b_5) = (b_2\,b_5\,b_4\,b_1\,b_3)$$

Encripte a mensagem $x = 01101\ \ 11011\ \ 11010\ \ 00110$ utilizando os cinco diferentes modos de operação: ECB, CBC, CFB, OFB e CTR. Forneça os correspondentes *ciphertexts* y.

Use $IV = 11001$ como vetor de inicialização.

Exercício: Recuperação de Vetor de Inicialização (IV) em AES-CBC

Em uma empresa, todos os arquivos enviados pela rede interna são automaticamente encriptados utilizando AES-128 no modo CBC. Uma *key* fixa é utilizada, e o vetor de inicialização (IV) é alterado uma vez por dia. A encriptação na rede é feita por arquivo, de modo que o IV é utilizado no início de cada arquivo.

Ao invadir o sistema, você conseguiu encontrar a *key* fixa do AES-128, mas não conhece o IV atual. Hoje, você conseguiu interceptar dois arquivos diferentes: um com conteúdo desconhecido, e outro conhecido por ser um arquivo temporário gerado automaticamente, contendo apenas o valor $0\times\texttt{FF}$.

Descreva brevemente como é possível obter o vetor de inicialização desconhecido (IV) e como você pode decifrar o conteúdo do arquivo desconhecido.

Exercício: Reutilização de IV no Modo OFB

O modo OFB (Output Feedback) utiliza o vetor de inicialização (IV) como ponto de partida para gerar a *keystream* usada na encriptação. Se o IV não for diferente a cada execução da operação de encriptação, a segurança do esquema é comprometida.

Dica: considere o uso de trechos conhecidos do *plaintext*, como *file signatures* (magic numbers).

Exercício: Tamanho do IV em AES-CTR

Estamos utilizando o AES no modo *counter* (CTR) para encriptar um disco rígido com capacidade de **1 TB**.

Qual é o comprimento máximo possível do vetor de inicialização (IV), considerando que cada bloco de 16 bytes deve ser encriptado com um valor único de contador?